EP30175 ④

• Nerot, Sebastien Charles
78580 Maule (FR)

(74) Representative: McCormack, Derek James et al
Motorola
European Intellectual Property Operations
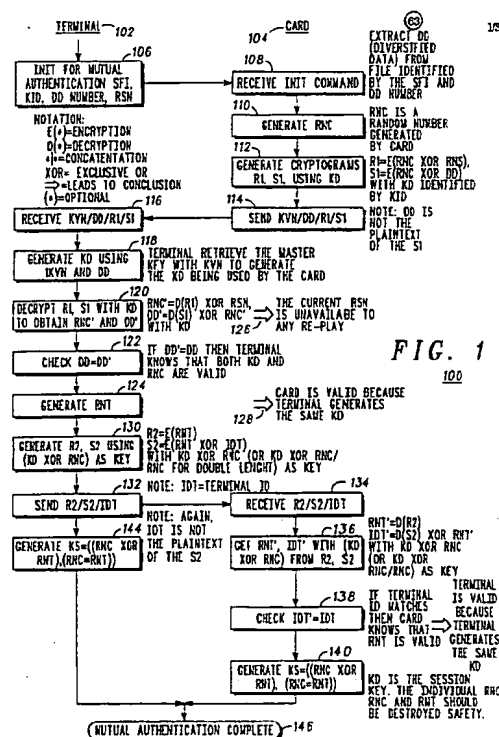Midpoint
Alencon Link
Basingstoke Hampshire RG21 7PL (GB)

(54) Transaction system, portable device, terminal and methods of transaction

(57) A method of mutually authenticating a transaction or a message between a Terminal (102) and a Smartcard (104), includes the steps of: transferring key data between a Terminal and a Smartcard; and generating a card key (Kd) at the Terminal (102) based on said key data wherein the generated card key (Kd) is equal to the card key of the Smartcard (104) thereby authenticating a valid Smartcard; and/or generating a Terminal identifier at the Smartcard (104) based on said key data wherein the generated Terminal identifier (IDt) is equal to the Terminal identifier (IDt) of the Terminal (102) thereby authenticating a valid Terminal (102).

A Smartcard, a Terminal (102), a Smartcard command set, a method of generating a session key, a dynamic session key, a set of instructions, a Commit command (352, 354) and a roll-back mechanism are provided.

This provides a high level of security, ensures data integrity and a fast commit processing and fast transaction time.

FIG. 1

EP 1 223 565 A1

BEST AVAILABLE COPY

Description

Field of the Invention

5   [0001]   This invention relates to a transaction between a portable device, such as a smart card, and a fixed device. The invention is applicable to, but not limited to, a secured transaction using a Contact-less Smartcard.

Background of the Invention

10   [0002]   In the field of this invention, namely secure and effective Smartcard transactions, it is important to insure the cryptographic authentication of both the portable device (Smartcard) and the ground (terminal).
As such, the concept of Mutual authentication between the portable device and the fixed device has been widely used in the Smartcard industry.
[0003]   Furthermore, it is known that the uniqueness and the non-reproducibility of the cryptographic key, as used to
15   certify the authenticity of the overall transaction, needs to be guaranteed. As such, the concept of Cryptographic Signature (cryptographic key) has been widely used in the Smartcard industry as a reliable means to authenticate messages or transactions.
[0004]   To complement this authentication process, a random session key, as opposed to a static key, is sometimes used to improve the security of the cryptographic operation involved. This makes a replay attack by a third party much
20   more difficult.
[0005]   A transaction is implemented as a critical section. The critical section commences, after a successful completion of the mutual authentication. The critical section is completed with a successful verification of the transaction signature. In such transactions, it is known that the integrity of the data stored in the portable device is an important design criterion.
25   [0006]   A Smartcard transaction involves more than one file. If a tear occurs in the middle of the transaction, some files might have been updated while other ones remain in the previous state. As a result the whole Smartcard file system is in an inconsistent state.
[0007]   For example, a transaction log file can have been updated while the internal electronic (E)-purse may not have been debited. For contact-less transactions between a Smartcard and a terminal, time is the most critical param-
30   eter.
[0008]   However, this approach has the disadvantages that Smartcard transactions require writing the transaction data twice. Prior art Smartcard transactions also require management of read and write flags. Such data management and control is suitable for contacted operations.
[0009]   However, such data management and control is unsuitable for a contact-less environment using EEPROM
35   non-volatile memory technology. Furthermore, such transaction management takes too much time in a Contact-less transaction, which typically must be completed in 200 msec.
[0010]   A further problem associated with mutual authentication is that such a technique is subject to crypto-analytic attack when the plain-text/cipher-text pair is available at the portable device/ground interface.
[0011]   Thus there exists a need in the field of the present invention to provide a Smartcard transaction mechanism,
40   a Smartcard/portable device, a Terminal and methods to enable a Smartcard transaction wherein the abovementioned disadvantages may be alleviated.

Statement of Invention

45   [0012]   In accordance with a first aspect of the present invention there is provided a method of mutually authenticating a transaction or a message between a Terminal and a Smartcard, as claimed in claim 1.
[0013]   In accordance with a second aspect of the present invention there is provided a Smartcard, as claimed in claim 11.
[0014]   In accordance with a third aspect of the present invention there is provided a Terminal, as claimed in claim 12.
50   [0015]   In accordance with a fourth aspect of the present invention there is provided a Smartcard command set, as claimed in claim 13.
[0016]   In accordance with a fifth aspect of the present invention there is provided a method of generating a session key to authenticate a transaction or a message, as claimed in claim 14.
[0017]   In accordance with a sixth aspect of the present invention there is provided a Smartcard, as claimed in claim 19.
55   [0018]   In accordance with a seventh aspect of the present invention there is provided a Terminal, as claimed in claim 20.
[0019]   In accordance with a eighth aspect of the present invention there is provided a dynamic session key of a transaction or a message, as claimed in claim 21.

2

[0020] In accordance with a ninth aspect of the present invention there is provided a set of instructions used in a transaction process between a Terminal and a Smartcard, as claimed in claim 22.

[0021] In accordance with a tenth aspect of the present invention there is provided a Commit command, as claimed in claim 25.

[0022] In accordance with an eleventh aspect of the present invention there is provided a roll-back mechanism, as claimed in claim 26.

[0023] In accordance with a twelfth aspect of the present invention there is provided an integrated circuit adapted for use in a Smartcard or terminal, as claimed in claim 29.

Brief Description of the Drawings

[0024] Exemplary embodiments of the present invention will now be described, with reference to the accompanying drawings, in which:

[0025] FIG. 1 shows a flowchart of a Terminal-Smartcard mutual authentication process in accordance with a preferred embodiment of the invention.

[0026] FIG. 2 shows a Smartcard Operating System Architecture in accordance with a preferred embodiment of the invention.

[0027] FIG. 3 shows a transaction session management process in accordance with a preferred embodiment of the invention.

[0028] FIG. 4 shows a transaction "Commit" management process in accordance with a preferred embodiment of the invention.

Description of Preferred Embodiments

[0029] In summary, the inventive concepts described herein propose, inter alia, a Smart Card command set and security architecture, preferably for use in Contact-less environments. The inventive concepts find particular applicability in Contact-less environments that are specifically mandated for speed of transaction.

[0030] The Smart Card command set and security architecture have applicability to public transport systems such as Train or Bus systems, as well as related fare or debit-based applications such as Parking and Taxis. The commands described are a combination of standard ISO7816 type commands as well as unique commands designed to ensure both security and transaction speed are accommodated in Contact-less type transactions.

[0031] Security, transaction integrity and transaction speed are of foremost importance in Contact-less applications. The preferred embodiment of the present invention describes a transaction architecture that tightly integrates all three of these attributes.

[0032] In particular, the inventors of the present invention have recognised that the mutual authentication, session key generation and beginning of critical section are distinct concepts that have not yet been merged in the field of Smartcard transactions. Furthermore, the signature of a transaction using data exchanged in both directions has not previously existed. In addition, the proposed mechanism includes the merging of the end-of-session with the authentication of the overall transaction.

[0033] A Mutual Authentication command (MAC) is an integration of Internal and External type authenticate commands such as described in ISO7816. This ensures authentication of both the Card and the Terminal during one command to speed transactions without compromising security. To eliminate the possibility of replay-type attacks, a dynamic Session Key is generated for every session (transaction) based on two random numbers.

[0034] For added security, and further to the concept of a session, a MAC is generated from all data passed throughout the transaction. This MAC is then verified again at the end of the transaction against the dynamic session key generated at the beginning of the transaction. This eliminates the possibility for a rogue terminal to take over the transaction following the authentication cycle.

[0035] Furthermore, to ensure system security, another file type and two unique commands are also defined. Separate Credit Purse and Debit Purse commands allow the system designer the flexibility to ensure that only a key that allows a 'reduction' of stored value amount (key assigned to Debit Purse) may be resident on buses and other transport equipment.

[0036] Such a feature is not possible using the Update Binary or Update Record commands of prior art mechanisms that do not regard direction of balance or value change. In addition, it is within the contemplation of the invention that the stored value purse may be pre-defined at personalization time with a maximum balance.

[0037] In accordance with one aspect of the preferred embodiment of the invention, to ensure transaction integrity a Rollback mechanism is employed on the card to predictably recover following a transaction tear.

[0038] The Commit command described herein is the final step in a transaction and accomplishes two things. First, it uses the session key to validate the transaction data passed throughout the transaction, as described above. Sec-

3

ondly, it commits the data to the card and makes this known to the terminal via a successful command response message.

[0039] If the terminal does not receive a successful response to the Commit command it may assume that a rollback will take place on the card at next power cycle, and the transaction data will not be processed on the card. This returns the card to the previous known valid state before the tear occurred.

[0040] A second less preferred alternative embodiment of the Commit/Roll Back mechanism is also provided. The alternative Commit/Roll Back mechanism is fast enough to be applicable to contact-less Smartcard applications.

[0041] In addition, an original management of record files and their pointers or indexes is used. Data of a file, which has to be protected against tear by this mechanism, can be updated in only one EEPROM write access in the best case. This mechanism is therefore more than twice the speed of prior art mechanisms.

[0042] Referring first to FIG. 1, a flowchart 100 indicating the Mutual Authentication and Session Key Generation methodology between a terminal 102 and a Smartcard 104 is provided, in accordance with the preferred embodiment of the invention.

[0043] The "Init for Mutual Authentication" command 106 is sent by the fixed terminal to the Smart card to start a transaction. The "Init for Mutual Authentication" command includes:

(i) The identifier of a file that contains diversification data DD. This identifier is sent in the form of an ISO 7816-4 5-bit short file identifier (SFI). The terminal knows the master key KID that is required to process the DD data from the card. This enables the same derived key Kd as the card to be built: Kd = f(MK, DD) where f represent any diversification algorithm and MK the master key identified by KID;

(ii) a data diversifier number (DD number) that contains the reference of the diversifier data within the diversifier file pointed to by SFI;

(iii) a KID, indicating which (Data Encryption Standard) (DES) key will be used. The card stores the diversified key Kd in the card applicable key file, the terminal stores the master key MK in a secure device; and

(iv) an eight-byte Random Sequential Number (RSN) to provide one-way identification.

[0044] In summary, the card has been loaded with a derived key 'Kd'. Derived key Kd is the result of the diversification of the master key (MK identified by its identifier KID) with the diversification data DD. This process is used as it is a security threat to store the same master cryptographic key in every smartcard.

[0045] The command is received 108 at Smartcard 104, where at least the diversifier data DD is extracted from the file identified by the SFI and DD number.

[0046] Preferably, the diversifiers are stored in fixed length record files of 8x8-byte record. On receipt of the "Init for Mutual Authentication" command, a RNC is generated by the Smart Card 110. The RNC ensures that the return message from Smartcard to the fixed terminal is valid only in this mutual authentication phase.

[0047] In addition, Cryptograms R1 and S1 are generated using Kd, as shown in step 112, with Kd identified by the master key from which it derives: KID. In other words, the card is forced to exercise its Kd to encrypt the message involved with RSN and RNC.

[0048] A session key is then agreed by both the terminal and Smartcard. Clearly, the session key (Ks) is generated such that it is impossible to determine a card key from the session key (Ks).

[0049] To avoid the potential danger of a third party searching a card from an exposed session, the session key (Ks) shall not be a function of the card key. In particular, this requirement is mandatory when a short card key such as single DES key, is used.

[0050] In the preferred embodiment of the invention, the "Init for Mutual Authentication" command 106 requires a 25-byte return message 114 from the Smartcard 104. In the 25-byte message 114, one byte is for the Key Version Number (KVN) that the Terminal 102 needs to identify the exact key being used in the Smartcard 104.

[0051] A further concatenated 8-bytes are used for the key diversifier (DD), specified by SFI, and the DD number. A further eight bytes are R1 = E(RNC ⊕ RSN) where the RNC is an 8-byte random number from the card and the E is an encryption algorithm such as DES. The last 8-bytes are S1 = Encrypted (RNC ⊕ DD).

[0052] The Terminal 102 receives the 25-byte message 116. The Terminal 102 generates the Kd value to be used by the Smartcard 104 by using the Master key MK, the received and extracted KVN and DD values. Once the Kd value has been obtained, the Terminal 102 is able to decrypt R1, S1. The Terminal 102 may then retrieve the

(i) RNC' by decrypting, with Kd, the cryptogram R1: RNC' = D(R1) ⊕ RSN, and

(ii) DD' by decrypting, with Kd, the cryptogram S1: DD'= D(S1) ⊕ RNC'.

[0053] Advantageously, the current RSN is then unavailable and any future replay of R1 and S1 will be detected, as shown in step 126.

[0054] If the retrieved DD' is determined to be equal to the DD in the received data, in step 122, then the Terminal 102 is confident that the Smartcard 104 (Kd and RNC) is valid.

[0055] The Terminal 102 generates an eight-byte random number RNT, as shown in step 124, and builds (RNC ⊕ Kd) that will be used as the key to encrypt R2, S2:

(i) R2 = encrypted RNT, using the key (RNC ⊕ Kd), and
(ii) S2 = encrypted (RNT ⊕ IDt), using the key (RNC ⊕ Kd). IDt = Terminal ID, as shown in step 130.

[0056] The Terminal 102 transmits the "Mutual Authentication" command including the following information to the Smartcard 104, in step 132:

(i) IDt, that is the terminal ID;

(ii) R2 = E(RNT) where E() indicates encrypted and the RNT is a random number from the terminal; and

(iii) S2 = E(RNT⊕IDt).

[0057] The "Mutual Authentication" is received at the Smartcard 104 in step 134. The Smartcard 104 uses the reverse processes to step 130, to obtain the RNT value as shown in step 136. The Smartcard 104 checks to see if the determined IDt' equals the received IDt, in step 138. If there is a match, the Smartcard 104 knows that the Terminal (RNT) is valid, as the Terminal 102 has generated the same Kd, as shown in step 142.

[0058] The Smartcard 104 generates the session key Ks, with the individual RNC and RNT values destroyed safely, as shown in step 140. At the same time, the Terminal 102 generates the same Ks, with the individual RNC and RNT values destroyed safely, as shown in step 144. The mutual authentication process, according to the preferred embodiment of the invention, is finished as shown in step 146.

[0059] If a single-length session key is required, in step 140 and step 144, Ks = (RNC ⊕ RNT). However, if a double-length session key is required, in step 140 and step 144, Ks = ((RNC ⊕ RNT), (RNC - RNT)).

[0060] The actual commands transferred between the Terminal 102 and Smartcard 104 in the mutual authentication process are shown in Tables 1 to 3.

Table 1:

Application Protocol Data Unit (C-APDU) Message 106 for Init for Mutual Authentication	
CLA	0x80
INS	0x7A
P1	SFI on the five most significant bits (msb) and DD number on the three least significant bits (lsb) P1=0x00 is reserved for personalisation only
P2	KID P2=0x00 is reserved for personalisation only
Lc	0x08
Data field	8 bytes for RSN
Le	0x19, 1 byte for KNV, 8 bytes for DD the diversifier, 8 byte for R1, 8 bytes for S1.

Table 2:

Response Application Protocol Data Unit (R-APDU) Message 114 for Init for Mutual Authentication	
Data field	KVN‖DD‖R1‖S1
SW1-SW2	Status bytes

Table 3:

Status words of response message.		
SW1	SW2	Meaning
0x90	0x00	Successful operation
0x6A	0x88	Key or Key file not found
0x69	0x82	Crypto verification failure
0x69	0x83	Secret blocked
0x69	0x84	Secret file invalidated
0x6E	0x00	CLASS not supported
0x6D	0x00	INS not supported
0x67	0x00	Wrong Lc or Le
0x6B	0x00	Incorrect P1 or P2
0x6A	0x82	File not found

[0061] The Terminal 102 sends the "Mutual Authentication" command 132 to the Smartcard 104 with two purposes. The first purpose is to acknowledge that the Smartcard 104 has been verified as a valid Smartcard 104 since the Terminal 102 generates the same card key according to KVN and diversifier from the Smartcard 104. The second purpose is to provide information about the Terminal 102 itself to the card to authenticate the Terminal 102.

[0062] The key for the above encryption is Kd $\oplus$ RNC.
Advantageously, the "Mutual Authentication" command does not require any return transmission other than a status word informing the terminal of the completion status of the operation.

[0063] If the status word is 0x6982, the card rejects the terminal: The key Kd is not authenticated and the card files, to which access is granted upon presentation of this key, remain locked.

Table 4:

Command Application Protocol Data Unit (C-APDU) message 132 for Mutual Authentication	
CLA	0x80
INS	0x7C
P1	0x00
P2	0x00
Lc	0x18
Data field	R2‖S2‖dt

Table 5:

Response Message 140.	
Data field	Empty
SW1-SW2	Status bytes

Table 6:

Status words of response message.		
SW1	SW2	Meaning
0x90	0x00	Successful operation
0x6A	0x88	Key or Key file not found

Table 6:   (continued)

Status words of response message.		
SW1	SW2	Meaning
0x69	0x82	Crypto verification failure
0x69	0x83	Secret blocked
0x69	0x84	Secret file invalidated
0x6E	0x00	CLASS not supported
0x6D	0x00	INS not supported
0x67	0x00	Wrong Lc
0x69	0x85	Command out of sequence
0x6B	0x00	Invalid P1 or P2

[0064]   A Debit Purse command message will decrease an E-purse current balance if the resulting new balance is greater than or equal to the predefined minimum balance. The message will include the short file identifier (SFI) of the E-purse file as well as the amount to be debited.

Table 7 -

Debit Purse command APDU **Message**	
CLA	'80'
INS	'56'
P1	See table 8
P2	'00'
Lc field	'04' bytes
Data field	Amount to be debited
Le field	'04'

Table 8 -

Coding of the P1					
	b8	b7	b6	b5	Meaning
b4	b3	b2	b1		
x	1 / x	0 / x	0 / x	x	P1: b8-b6 = (100)b, then b5-b1 are a short EF identifier, exclude (11111) b . If b5-b1 are all 0, then use current file.

7

Table 9 - Debit Purse response APDU Message

Data field SW1-SW2	4 bytes of Updated Balance Status bytes

SW1	SW2	Meaning
'90'	'00'	Successful operation
'64'	'00'	Memory unchanged
'65'	'81'	Memory failure (unsuccessful updating).
'67'	'00'	Wrong length (wrong Lc and/or Le field).
'69'	'81'	Command incompatible with file structure.
	'82'	Security status not satisfied.
	'84'	Referenced data invalidated*
	'89'	Selected file blocked
'6A'	'82'	File not found.
'6B'	'00'	Wrong parameters (offset outside the EF).

[0065]   The updated balance is less than the predefined minimum balance allowed.

[0066]   A Reverse Debit command message will increase the E-purse current balance, when the amount to be credited is less than or equal to the amount previously debited in the last transaction. The message will include the short file identifier (SFI) of the E-purse file and the amount to be credited.

Table 10 -

Reverse Debit command APDU message	
CLA	'80'
INS	'58'
P1	See table 7
P2	'00'
Lc field	'04' bytes
Data field	Amount to be credited
Le field	'04'

122356541 1 3

Table 11 – Reverse Debit response APDU message

Data field SW1-SW2	4 bytes of Updated Balance Status bytes

SW1	SW2	Meaning
'90'	'00'	Successful operation
'64'	'00'	Memory unchanged
'65'	'81'	Memory failure (unsuccessful updating).
'67'	'00'	Wrong length (wrong Lc and/or Le field).
'69'	'81' '82' '84' '89'	Command incompatible with file structure. Security status not satisfied. Referenced data invalidated* Selected file blocked
'6A'	'82'	File not found.
'6B'	'00'	Wrong parameters (offset outside the EF).

[0067]   The amount to be credited is more than the amount previously debited in the last transaction.

[0068]   The Commit command message will allow the card to commit all the data it has updated during the session. The Commit will contain an eight-byte MAC over all bytes exchanged during the session.

Table 12 -.

Commit command APDU message	
CLA	'80'
INS	'78'
P1	'00'
P2	'00'
Lc field	'08'
Data field	SACT
Le	Empty

Table 13 - Commit response APDU message

Data field SW1-SW2	Empty Status bytes

SW1	SW2	Meaning
'90'	'00'	Successful operation
'65'	'81'	Memory failure
'67'	'00'	Wrong length (empty or wrong Lc field).
'69'	'85'	Conditions of use not satisfied (Command out of sequence)
'69'	'94'	SACT failed

Session key generation

[0069]   The attached flow chart shows that the session key Ks = (RNC⊕RNT, RNC-RNT) can be obtained on both the Fixed Terminal 102 and the Smartcard 104.

[0070]   To use this protocol, a constant may be agreed for the DD beforehand.

[0071]   For example, the DD may be the ASCII string of the date carrying personalisation. Alternatively, the DD may just be a constant such as 0x0102030405060708. When the card reads "P1=0" and "P2=0", it will use the agreed value as DD to carry out the protocol.

[0072]   In the preferred embodiment of the invention, Kd is used to securely exchange RNC and RNT. A successful "Init_for Mutual Authentication/ Mutual Authentication" message has three effects:

(i) Kd key is authenticated;

(ii) a session Key Ks, that has no connection with Kd is built (Ks depends only on RNC and RNT); and

(iii) the critical section signal is sent to the Smartcard core operating system.

[0073]   Advantageously, with regard to the security in setting up and mutually authenticating the transaction, it is difficult to search Kd with R1, S1 and DD as a third party has to guess the RNC being used.

[0074]   As a result, the on-board random number generator enhances the security of the protocol in which a single length key cryptosystem, such as DES, is used.

[0075]   Furthermore, the process is beneficially arranged such that disclosure of the session key Ks may not be exploited to attack the card key Kd.

[0076]   As an additional fail-safe mechanism, the scheme still works when the card random number generator (RNG) fails. However, in this case, the system's security will solely rely on the strength of the underlying encryption system.

[0077]   A random sequential number RSN is sent to the card from step 106 to resist possible replay attack. Any replay of R1 and S1 can be detected at step 122.

[0078]   Advantageously, the Smartcard 104 is verified by the Terminal 102 at step 122 and the Terminal 102 is verified

by the Smartcard 104 at step 138.

[0079] In accordance with a preferred embodiment of the present invention, a portable device (Smartcard) Operating System (OS) Architecture 200 is also described, as shown in relation to FIG. 2. The Operating System Architecture 200 is divided into distinct areas:

(i) the OS Core 202; and
(ii) the Smartcard OS Applications 204.

[0080] The OS Core 202 includes hardware resources 206, that includes a processing unit 208, coupled to a Non-Volatile Random Access Memory (NV-RAM), Random Access Memory (RAM) and read-only memory (ROM) 210, coupled to the crypto engine 212.

[0081] The OS Core 202 also includes hardware interfaces 214, that includes a physically-contacted interface 216 and a physically-contactless interface 218. In addition, the OS Core 202 includes platform low-level drivers 220 having transmission protocol 224 and anti-collision functions 226. Platform middleware level (card application) services 230 are also provided, including an application manager and interface 232 and ISO secure messaging layer 234.

[0082] In summary, the aforementioned Smartcard configuration forms the card OS core and provides the following services:

(i) communication: communication driver and protocol stack; and

(ii) middleware: made of an optional message ciphering/deciphering layer and a mandatory (optional) message routing that direct the data to/from the proper application to/from the terminal.

[0083] The Smartcard OS Applications 204, have a platform native application (Card management) function 240. This platform native application function 240 includes Init 242, EMV 244, easy entry 246 and admin 248. Custom applications (product specific) functions 250, including E-purse 252 and Transit 254 are provided. In these applications the high level APDU, as described above, are processed.

[0084] As an example: the C-APDU 'INIT_FOR_MA' arrives through the Contact-less interface. The transmission protocol stack de-capsulates the command and the application manager sends the relevant data to the application in control. The application processes the data and sends its response through the same channel.

[0085] The portable device (Smartcard) Operating System 200 ensures the administration and the security of the secured file system mapped in the Non-Volatile Random Access Memory (NV-RAM) 210. The Operating System Core features the services used by the various applications that are implemented on top of the OS core and use its resources.

Session Details.

[0086] A transaction using a portable device is basically a succession of elementary read and write operations/instructions on data stored in the NV-RAM 210 of the Smartcard. In order to avoid any inconsistency in the data stored in the NV-RAM 210 of the Smartcard, it is important that these elementary read and write operations are either all performed ("Committed") or not performed at all ("Roll-Backed").

[0087] As an example: if the transaction is to debit a value and log this debit, it would not be acceptable that the debit operation is performed without it being logged.

[0088] The functional operation between the start of the transaction and the end of the transaction is generally termed the 'Critical Section'. All modifications of the NV-RAM data performed during the critical section must be either committed atomically after reception of a valid "Commit" or disregarded.

[0089] Referring now to FIG. 3, a flowchart 300 of a Session Management data flow between a fixed terminal 302 and a Smartcard portable device 304 is shown. The mutual authentication process 306 is performed, as described earlier with reference to FIG. 1.

[0090] The mutual authentication process includes the Terminal 302 transmitting the "Init for Mutual Authentication" message 308 to the Smartcard portable device 304, which responds with the 25-byte return message 310 from the Smartcard portable device 304.

[0091] The "Mutual Authentication" message 312 is then sent from the Terminal 302 to the Smartcard portable device 304, which responds 314 to complete the mutual authentication process. This process ensures that a session key is built by both the Terminal 302 and the Smartcard portable device 304, using the random numbers exchanged during the authentication process.

[0092] The critical section is commenced by a BEGIN CRITICAL SECTION signal sent to the OS core of the Smartcard portable device 304 by one of its embedded applications 240,242,244,246,248,250 or 254 after a successful mutual authentication pass. During the critical section, the data directed to the portable file system, as shown in mes-

sages "Session Commands 1-4 322, 326, 330, 334 are written into cache buffers and a session authentication code (SAC) is computed of the data sent and received from/to the Smartcard portable device 304.

[0093] The SAC computed at the Terminal 302, as calculated from all data sent to, and all data received from, the Smartcard portable device 304 is termed a SACT. The SAC computed with the same data on the Smartcard portable device 304 side is termed a SACC.

[0094] The algorithm to compute the SAC depends on the cryptographic capabilities of the involved device. It may be, for example, a Chained (CBC) DES using the session key. The purpose of this SAC is to compute a signature of the exchanged data.

[0095] The session is terminated upon reception and successful verification of the SACT as transmitted 352 from the fixed Terminal 302 to the Smartcard portable device 304. The Smartcard portable device 304 compares the incoming SACT with its internally computed SACC. The comparison status id is then sent back 354 to the Terminal 302.

[0096] If there is a match, the Smartcard portable device 304 and the Terminal 302 have completed an authenticated transaction and the Smartcard portable device 304 initiates its commit/end of session algorithm 350. If there is no match, the portable device rejects the transaction. The data in the NV-RAM cache buffer are disregarded and zeroed and the transaction is cancelled.

Commit and Tear Recover Algorithms

[0097] The portable device Operating System features a mechanism that ensures the atomicity of NV-RAM data update operations. This mechanism is built upon internal resources of the Operating System core used by the native or product-specific applications. These internal resources include:

(i) Internal request to notify the portable device operating system core of the beginning of a critical section: 'BEGIN CRITICAL SECTION';

(ii) Internal request to notify the portable device operating system core of the ending of the critical section followed by actual data update: 'COMMIT';

(iii) Internal request to notify the portable device operating system core of the ending of the critical section followed by data update cancellation: 'ROLLBACK' ;

(iv) A check algorithm at portable device boot-up time;

(v) A RAM cache buffer; and

(vi) A Non-Volatile Memory cache buffer (NV-RAM).

[0098] Referring now to FIG. 4, a Commit Algorithm according to a preferred embodiment of the present invention is shown. Initially, both the RAM Cache 404 and NV-RAM cache 406 are empty and zeroed 420, 422. The data flow during a critical session includes the Smartcard portable device Operating System intercepting each incoming data 402 that impacts the NV-RAM file system. The OS encapsulates the data into a controlling structure named Anti Tear Data Structure (ATDS). These structures are stored into the RAM cache, as shown in data flow 430.

[0099] When the RAM cache 404 is full, its content is copied 432 into the NV-RAM cache buffer 406. The RAM cache is then zeroed. The critical section is ended by the reception of the END CRITICAL SECTION signal associated with a COMMIT or ROLLBACK directive.

[0100] This RAM cache has been introduced to cope with the fact that the NV-RAM (usually EEPROM) is updated by block: N bytes of data can and is preferably written in one single programming operation. Accumulating data into the RAM cache until N-bytes are ready to be programmed minimises the number of update operations and therefore minimises the total transaction time. This algorithm is a major contribution to the optimisation of the transaction time.

[0101] The portable device Operating System processes the COMMIT directive by copying 440 the ATDS that are in the RAM cache 404 into the EEPROM cache 406. The RAM cache is then zeroed.

[0102] The commit is carried out i.e. the file system is updated with the ATDS data, as shown in data flow 442. The EEPROM cache 406 is then erased. The execution of the ROLLBACK directive is performed by erasing the RAM cache 404 and EEPROM cache 406.

Anti-Tear Algorithm

[0103] When the portable device boots up, the content of the NV-RAM scratch buffer is checked to see whether or

not the file system was entirely updated during the last commit process. If the previous commit has been aborted, the Operating System continues to commit before accepting any other request. The portable device operating system zeros the NV-RAM cache and is ready to accept any request.

[0104] In the less-preferred, alternative embodiment of the present invention, a rollback (sometimes termed 'tear recovery') mechanism is provided. The 'tear recovery' or 'rollback' is a means used to ensure that the data stored in the Smartcard would return to a stable state after an unexpected event, such as power off during a transaction. The data on the card might be stored in one or many ISO 7816-4 compliant elementary file(s).

[0105] As mentioned earlier, a new roll back mechanism is needed that reduces the time needed for the data stored in the Smartcard to return to a stable state, particularly for the Contact-less Smartcard environment. In the preferred embodiment of the invention, this card will have an ISO 7816-4 compliant file system.

[0106] Data where modifications must be secured by a tear recovery mechanism must be stored in 7816-4 record files. These files can be linear fixed or cyclic, depending on the card design option. Furthermore, the files should have more than one record (actual number plus one).

[0107] In the preferred embodiment of the present invention, namely that of ISO 7816-4, record files cannot contain more than 254 records. A cyclic file is a linear fixed record file with a specific pointer management. A pointer named "last record pointer" which indicates the last updated record is stored in the file header.

[0108] The data record structure is shown in Table 14:

Table 14:

Counter	Transaction Id	Data field
1 byte	2 bytes or less	N bytes

[0109] The 'transaction Id' is a number provided by the terminal to select and stamp the transaction. A transaction session begins with the ISO 14443 card selection and ends with the card power off. The generation of this id is explained below. This number is associated with only one transaction scheme.

[0110] In the case of a cyclic file, the 'counter' byte is used to secure a tear, which occurs after the record update and before the update of the 'last record pointer' in the file header. The 'counter' byte may also be a "pointer" of the last updated record in the case of linear fixed file implementation, which could lead to the creation of a new type of cyclic record file. This byte is preferably used as an unsigned value.

[0111] A file named 'links file' features the Id of all the files involved in the transaction. All these files must be modified together or remain together in the initial state. It is an ISO 7816-4 record file containing one or more record. This file can be linear fixed or cyclic depending on the card design option.

[0112] Each record has a length depending on the maximum number of files involved, as shown in Table 15:

Table 15:

Last transaction flag	Number of Id	File ID #1	File Id #2	...	File Id #n	Transaction Id
1 byte	1 byte	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes

where:

(i) The 'last transaction flag' is set when the transaction starts and is cleared after a complete and successful transaction. It is the transaction tear indicator;

(ii) The 'number of Id' is the number of files secured by the tear recovery mechanism associated with the Transaction. This number cannot be greater than a value fixed by the card's or the terminal's application design; and

(iii) The Ids of each involved file(s) in this secured transaction follows the 'Number of Id' field.

[0113]   It is within the contemplation of the invention that the 'Transaction id', or the sequence of files involved in the transaction, may be managed by the terminal. In this case the 'links file' may be used to store the 'last transaction flag' only or a part of the information described above.

[0114]   In the preferred embodiment of this aspect of the invention, the card design may be configured in a number of ways. The preferred configuration is as follows.

Transaction Id:

[0115]   There are numerous ways to generate a Transaction Id, the preferred options being:

(i) generated dynamically at the beginning of the transaction by the terminal; and

(ii) registered definitively at the creation of the associated record of the 'links file'.

[0116]   The Transaction Id length may be one or two bytes depending on the Tear recovery configuration. The preferred four ways of designing a Tear recovery mechanism use record files according to the combination of two 'links file' options and two data file options:

(i) The 'links file' options:

[0117]

(ia) The 'links file' may be designed once, with as many records as the number of possible transaction schemes. If there is no more than 255 transactions scheme, one byte length 'Transaction Id' is enough. If there is more than 255 transactions scheme, two bytes length 'Transaction Id' must be used.
Each record's 'Transaction Id' is written once. It is within the contemplation of the invention that the transaction scheme of one or more record may be updated by the terminal 'on-the-fly'.

(ib) The 'links file' may be updated dynamically with a transaction Id generated by the terminal. The terminal may use only one record in case of single record file. It may use more than one record in a cyclic file. In this case, the terminal has to write a record corresponding to the new transaction scheme with the files involved (useful to trace previous transactions).

(ii) The data files may be cyclic or linear fixed record files.

[0118]

(iia) Linear fixed record files case:
In this case, the record, which contains the highest 'counter' value, is the last updated record. This highest 'counter' value is selected as follows:
If the record N has a 'counter' value of M and the next one (N+1) has a 'counter' value of M+1, the record N+1 clearly has the current highest 'counter' value. The same selection continues with the record N+1.

[0119]   If the record N has a 'counter' value of M and the next one has a 'counter' value NOT equal to M+1, the record N clearly has the highest 'counter' value. It is then the last updated record. The tear recovery mechanism reads the last updated record. It then increments its 'counter' value (now M+1) and modifies its data field. It then updates the next record. If the next record doesn't exist, the first record of the file is updated, as shown in Table 16.

Table 16:

Record N -1 (counter value = M -1)

This is the last record ->

Record N (counter value = M)

This the record to update ->

Record N+1 (counter value = M+1)

[0120]   To further explain this operation, let us consider a file that contains ten records, with a last 'counter' value of "56". The structure will be as shown in Table 17 below:

Table 17:

	Record 1 (counter value = 51)
	Record 2 (counter value = 52)
	Record 3 (counter value = 53)
	Record 4 (counter value = 54)
	Record 5 (counter value = 55)
This is the last record ->	Record 6 (counter value = **56**)
	Record 7 (counter value = **47**)
	Record 8 (counter value = 48)
	Record 9 (counter value = 49)
	Record 10 (counter value != 50)

[0121] Thus the record 6 is the last updated record. The new record will overwrite the record 7 with a 'counter' value of 57. The result is shown in Table 18 below:

Table 18:

Record 1 (counter value = 51)
Record 2 (counter value = 52)
Record 3 (counter value = 53)
Record 4 (counter value = 54)
Record 5 (counter value = 55)
Record 6 (counter value = **56**)
Record 7 (counter value = **57**)
Record 8 (counter value = 48)
Record 9 (counter value = 49)
Record 10 (counter value != 50)

This is the updated record -> (points to Record 7)

[0122]   Advantageously, file updating uses only one 'write' instruction. However, it must read N records to select the last updated record.

(iib) Cyclic record files case:
In this case, the tear recovery mechanism updated the next record. It takes two 'write' instructions per file to update. The first write instruction is performed for record update. The second write instruction is performed for the 'last record pointer' update in the file header. Advantageously, no read record is performed.

[0123]   To clarify the Tear recovery mechanism in accordance with this aspect of the invention, let us first consider the nominal case where the previous transactions were all conducted successfully. All of the 'last transaction flag(s)' are therefore cleared.

[0124]   If the files involved in the transaction are known in advance, the appropriate record is selected in the 'links file'. If the files involved in the transaction are not known in advance, a new record in the 'links file' is dynamically created by a terminal "CREATE_RECORD" command.

[0125]   In both cases, the 'last transaction flag' of the record is set to indicate that the transaction is in progress and

the 'transaction id' is set or updated to the current Transaction Id if the terminal has to generate it. Files involved in the transaction (registered in the current record of the 'links file') should be modified.

[0126]   A new record containing the new data (see record structure described above) is created with its 'transaction Id' field set to the current Transaction Id. The new record's 'counter' field set to the value of previous record plus one. At the end of the transaction, the 'last transaction flag' of the 'links file' is cleared to indicate that the transaction has completed successfully.

[0127]   Let us now consider a "Recovery" case, where a previous transaction has not successfully completed. If a tear appears before the 'Last transaction flag' update, no recovery is needed. This is due to the fact that the last transaction has not had enough time to perform any operation.

[0128]   If a tear appears after the 'Last transaction flag' update has occurred a recovery operation is needed. One 'links file' record has its 'Last transaction flag' set. The recovery is finished if the record is inconsistent (namely if (i) a file Id does not correspond to an existing file or (ii) if the number of the file Id is greater than allowed). An error is returned by the card if the OS is designed to support this tear recovery mechanism.

[0129]   For each file registered in this record, the terminal or the Smartcard application looks for a current record with a 'transaction Id' equal to the 'links record' 'transaction Id' field. If there is a match, and if the file is cyclic, the 'last record pointer' of this file is updated to point on the previous record, which contains the previous data. The roll back is implemented as a file pointer update.

[0130]   If there is a match, and if the file is a linear fixed file, the last updated record 'counter' value is set to the oldest record 'counter' value that is in the file, minus one. To further explain this operation, let us consider a file that contains 10 records, with a last 'counter' value of 56. The structure will be as shown in Table 19 below:

Table 19:

Record 1 (counter value = 51)
Record 2 (counter value = 52)
Record 3 (counter value = 53)
Record 4 (counter value = 54)
Record 5 (counter value = **55**)
Record 6 (counter value = **56**)
Record 7 (counter value = **47**)
Record 8 (counter value = 48)
Record 9 (counter value = 49)
Record 10 (counter value != 50)

Last record -> (points to Record 6)

[0131] Thus, the record "6" is the last updated record. A rollback has to be performed. The new 'counter' value of record 6 is updated to the oldest record, namely (47) minus one (46). Therefore, the record 5 becomes the 'last record'.

[0132] The result is:

5

10

15

Rolled back record ->

20

25

30

35

Record 1 (counter value = 51)	
Record 2 (counter value = 52)	
Record 3 (counter value = 53)	
Record 4 (counter value = 54)	
Record 5 (counter value = 55)	
Record 6 (counter value = 46)	
Record 7 (counter value = 47)	
Record 8 (counter value = 48)	
Record 9 (counter value = 49)	
Record 10 (counter value != 50)	

40

[0133]   The 'Last Transaction' flag of the 'links file' is then cleared to reflect a stable state.

[0134]   It is noteworthy that a power off during the roll back process cannot affect the stability of the tear recovery system as the current record of the rolled back file does not have the same 'Transaction Id'. Hence, it cannot be rolled back anymore. The tear recovery will continue until the previous stable state is reached.

45 [0135]   To quantify the benefits provided by the above alternative roll-back mechanism, the following results are provided. The following timings are computed for a ST16RF48 in Contact-less mode with an external radio frequency (RF) signal of 13.56 MHz.

[0136]   In the case of a dynamic 'Transaction Id' generation, there is $15*10^{-6}$ chance of failure for a 16-bit id. There is less than $2.5*10^{-10}$ chance of failure for a 32-bit id (in the case where 16 bits are not enough). In the case of static 50 'Transaction Id' attribution (one time transaction scheme registration), there is no chance of failure.

[0137]   The performance of the tear recovery mechanism may be assessed according to the four alternative implementations described above:

(i) Data file is a Cyclic record file and the Tear recovery mechanism is carried out in the Terminal management.
55 This is the best case if the involved cyclic record files have more than 4 records. It takes 2 write EEPROM accesses per file (typically 20 ms). It can be implemented using only ISO commands.

(ii) Data file is a Linear fixed record file and the Tear recovery mechanism is carried out in the Terminal management.

It takes one write EEPROM access per file (typically 10 msec's) and needs as many read commands as record numbers at most. It can be implemented using only ISO command. Because of the read command delay (more than two msec's per read command including transmission time at 105.9 Kbps), this case is not viable if the involved record files have more than four records.

(iii) Data file is a Cyclic record file and the Tear recovery mechanism is carried out in the Card management. This option takes two write EEPROM accesses per file (typically 20 msec).

(iv) Data file is a Linear fixed record file and the Tear recovery mechanism is carried out in the Card management. This option is the fastest because it takes about 5000 cycles at most to compute an internal record selection (three msec for 254 record, 130 μsec for 11 records). It needs only one write EEPROM access (typically 10 msec).

[0138]   In the case where N files are involved in the current transaction scheme, and a tear appears after M files update (M < N), the number of 'write' instructions for recovery and transaction completion, for each of the four implementation options (above) are:

(a) Recovery, for options (i) and (iii) requires ((M*2)+1) write access instructions;

(b) Recovery, for options (ii) and (iv) requires (M+1) write access instructions;

(c) Transaction completion, for options (i) and (iii) requires ((N*2)+2) write access instructions; and

(d) Transaction completion, for options (i) and (iii) requires (N+2) write access instructions.

[0139]   Hence, in this particular case, option (iv) appears to be the most efficient. However, this case needs an OS specific tear recovery feature.
[0140]   Applying the above equations to real-life situations results in the following. Let us consider the situation where three files are involved in the current transaction scheme, and a tear appears after two file updates:

(a) Recovery, for options (i) and (iii) requires five write access instructions;

(b) Recovery, for options (ii) and (iv) requires three write access instructions;

(c) Transaction completion, for options (i) and (iii) requires eight write access instructions; and

(d) Transaction completion, for options (i) and (iii) requires five write access instructions.

[0141]   Alternatively, for the situation where five files are involved in the current transaction scheme, and a tear appears after three file updates:

(a) Recovery, for options (i) and (iii) requires seven write access instructions;

(b) Recovery, for options (ii) and (iv) requires four write access instructions;

(c) Transaction completion, for options (i) and (iii) requires twelve write access instructions; and

(d) Transaction completion, for options (i) and (iii) requires seven write access instructions.

[0142]   It will be understood that the transaction system, Smartcard/portable device, Terminal and methods of performing a transaction, as described above, provide the following advantages:

(i) provides a high level of security as the transaction starts with a robust MUTUAL_AUTH and ends with a COMMIT signal sent with a cryptographic signature of all data using a random session key;

(ii) data integrity is ensured by a use of a critical section mechanism. This mechanism has been designed to ensure a fast commit processing time;

(iii) a fast transaction time is ensured by (a) merging the initial MUTUAL_AUTH with the session key generation

with the beginning of the critical section; and (b) merging the end of the transaction with the cryptographic authentication of the transaction;

(iv) the mechanism requires limited proprietary commands, namely two new commands (INIT_FOR_MA and MA) at the beginning of the session and the introduction of one new command (COMMIT) at the end of the session;

(v) the mechanism exchanges well known ISO/IEC 7816-4 commands between the Smartcard device and the terminal during the session, thereby significantly limiting the complexity of the solution; and

(vi) the optimised roll back mechanism performs at more than twice the speed of prior art mechanisms.

[0143]   It will, of course, be understood that a smart card or terminal as described above will typically be constructed around one or more integrated circuits that are adapted to provide the required functionality described above.
[0144]   Thus, a transaction system, Smartcard/portable device, Terminal and methods of performing a transaction have been provided whereby the problems associated with prior art arrangements have been substantially alleviated.

Claims

1.   A method of mutually authenticating a transaction or a message between a Terminal (102) and a Smartcard (104), the mutual authentication method **characterised by** the steps of:

transferring key data between a Terminal and a Smartcard; and
generating a card key (Kd) at the Terminal (102) based on said key data wherein the generated card key (Kd) is equal to the card key of the Smartcard (104) thereby authenticating a valid Smartcard;
and/or
generating a Terminal identifier at the Smartcard (104) based on said key data wherein the generated Terminal identifier (IDt) is equal to the Terminal identifier (IDt) of the Terminal (102) thereby authenticating a valid Terminal (102).

2.   The method of mutually authenticating a transaction or a message between a Terminal (102) and a Smartcard (104) according to claim 1, the mutual authentication method further **characterised by** the steps of:

sending (106) a mutual authentication initiation message including a master key identifier (KID) and diversification data identifiers (DD number, SFI) from the Terminal (102) to the Smartcard (104);
processing (108, 110) the mutual authentication initiation message including the master key identifier (KID) and diversification data identifier (DD number, SFO) at the Smartcard (104);
sending (114) a return message from the Smartcard (104) wherein the message includes a card Key version number (KVN);
receiving and processing (116) the return message at the Terminal (102);
extracting the card Key version number (KVN) used in the Smartcard (104) from the return message;
deriving (118) Master key (MK) using retrieved diversification data (DD') used in the Smartcard (104) at the Terminal (102); and
comparing (122) retrieved diversification data (DD') to diversification data (DD) in the received data, to determine if the Smartcard (104) is valid.

3.   The method of mutually authenticating a transaction or a message between a Terminal (102) and a Smartcard (104) according to claim 1 or claim 2, the mutual authentication method further **characterised by** the steps of:

transmitting (132) a Mutual Authentication command from the Terminal to the Smartcard (104) wherein the Mutual Authentication command includes a Terminal identifier (IDt) ;
receiving (134) the Mutual Authentication command at the Smartcard (104);
deriving (118) a retrieved Terminal identifier (IDt') used in the Terminal (102) at the Smartcard (104); and
comparing (138) the derived Terminal identifier (IDt') to the received Terminal identifier (IDt), at the Smartcard (104) to determine if the Terminal (102) is valid.

4.   The method of mutually authenticating a transaction or a message according to claim 1 or claim 2, the method further **characterised by** the step of:

generating (110) a random number (RNC) at the Smart Card in order to ensure that the return message from Smartcard to the fixed terminal is valid only in the mutual authentication phase.

5.   The method of mutually authenticating a transaction or a message according to any one of the preceding claims, the method further **characterised by** the step of:

generating (112) at least one cryptogram (R1, S1) using the first Smartcard derived key (Kd) identified by KID, thereby encrypting the return message (114).

6.   The method of mutually authenticating a transaction or a message according to any of preceding claims 1 to 3, wherein the encrypted return message (114) is encrypted using a pre-determined random sequence number generated by the Terminal (102) and a random number generated by the Smartcard (104).

7.   The method of mutually authenticating a transaction or a message according to any one of preceding claims 1 to 3 or claim 6, the method further **characterised by** the step of:

deriving (118) the master key (MK) at the Terminal (102) using the card key and the diversification (DD) received from the Smartcard (104);
decrypting (112), at the Terminal (102) at least one cryptogram (R1, S1) using the Terminal extracted second derived key (Kd).

8.   The method of mutually authenticating a transaction or a message according to any of the preceding claims, the method further **characterised by** the step of:

retrieving a random number generated by the card (RNC') by applying the at least one decrypted cryptogram and a data diversification (DD) of the Smartcard (104).

9.   The method of mutually authenticating a transaction or a message according to any of the preceding claims, wherein the steps of sending an initial mutual authentication message and sending a mutual authentication achieve at least one of: external authentication, internal authentication, session key creation and a beginning of critical section.

10.   The method of mutually authenticating a transaction or a message according to any of the preceding claims, wherein the Terminal (102) provides information about the Terminal (102) to the Smartcard (104) to authenticate the terminal (102).

11.   A Smartcard (104) adapted to perform any of the method steps of mutually authenticating a transaction or a message according to any of the preceding claims.

12.   A Terminal (102) adapted to perform any of the method steps of mutually authenticating a transaction or a message according to any of the preceding claims 1 to 10.

13.   A Smartcard command set including the initiating mutual authentication command and mutually authentication command of any one of the preceding claims.

14.   A method of generating a session key to authenticate a transaction or a message, in particular according to the method of mutually authenticating a transaction or a message of any of the preceding claims, the method **characterised by** the steps of:

agreeing (146) a session key by both a Terminal (102) and a Smartcard (104), such that a card key cannot be generated from the session key.

15.   The method of generating a session key to authenticate a transaction or a message according to claim 14, wherein the method is further **characterised by** the steps of:

generating (140, 144) the session key (Ks), by both the Smartcard (104) and the Terminal (102), using a random number (RNC) generated by the Smartcard (104) and a random number (RNT) generated by the Terminal (102).

16. The method of generating a session key to authenticate a transaction or a message according to claim 14 or claim 15, wherein the method is further **characterised by** the step of:

    destroying the generated random numbers (RNC, RNT) prior to agreeing (146) a session key (Ks).

17. The method of generating a session key to authenticate a transaction or a message according to any one of preceding claims 14 to 16, wherein the session key is used to validate transaction data transferred during a trans-action.

18. The method of generating a session key to authenticate a transaction or a message according to any one of preceding claims 14 to 17, wherein the session key is generated after a successful authentication process, the method further **characterised by** the steps of:

    computing a session authentication code (SAC) by encrypting with session key (Ks) the data sent and received during the transaction or message transfer to avoid replay attack.

19. A Smartcard (104) adapted to perform any of the method steps of generating a session key to authenticate a transaction or a message according to any of the preceding claims 14 to 18.

20. A Terminal (102) adapted to perform any of the method steps of generating a session key to authenticate a trans-action or a message according to any of the preceding claims 14 to 18.

21. A dynamic session key of a transaction or a message contained within a mutual authentication method of any of preceding claims 1 to 10.

22. A set of instructions used in a transaction process between a Terminal (102) and a Smartcard (104), wherein the set of instructions is **characterised by** distinct credit purse and debit purse values to ensure that only a key used in the transaction process that allows a reduction of a stored amount is resident on the Terminal (102) for a debiting process.

23. The set of instructions used in a transaction process between a Terminal (102) and a Smartcard (104), according to claim 22, further **characterised by** the key used in the transaction process allowing a reduction of a stored amount from the Terminal (102) to the Smartcard (102) in the debiting process.

24. The set of instructions used in a transaction process between a Terminal (102) and a Smartcard (104), according to claim 22 or claim 23, further **characterised by** the credit purse and debit purse values being pre-defined or updated at a personalisation time of the Smartcard (104), preferably with a maximum balance.

25. A Commit command (352, 354) for use as a final step in a transaction process between a Terminal (102) and a Smartcard (104), the Commit command **characterised in** it uses a session key to validate transaction data passed during a transaction, and it commits data to the Smartcard (104), making such committed data known to the Ter-minal (102) via a successful command response message.

26. A roll-back mechanism, in a transaction process between a Smartcard (104) having at least one random access memory cache buffer and at least one non-volatile random access memory cache buffer, in particular using the Commit command of claim 25, wherein the roll-back mechanism uses random access memory cache buffers to accumulate transaction or message data prior to completing the transaction or message by transferring the accu-mulated transaction or message data into a file system contained within the Smartcard(104) or Terminal (102), in order to minimise the number of random access memory updates.

27. The roll-back mechanism in a transaction process between a Smartcard (104) according to claim 26, wherein the roll-back mechanism is automatically enabled after a successful mutual authentication.

28. A method of mutually authenticating a transaction or a message between a Terminal (102) and a Smartcard (10.. according to claim 1, or a Smartcard (104) according to claim 11 or claim 19, or a Terminal (102) according to clai 12 or claim 20, or a Smartcard command set according to claim 13, or a method of generating a session key authenticate a transaction or a message according to claim 14 a dynamic session key of a transaction or a messa; according to claims 21, or a set of instructions used in a transaction process between a Terminal (102) and

24

Smartcard (104) according to claim 22, or a Commit command (352, 354) according to claim 25, or a roll-back mechanism according to claim 26, when used in a Contact-less transaction environment.

29. An integrated circuit adapted for use in a Smartcard or Terminal to perform according to any one of preceding claims 11, 12, 19 or 20.

5

10

15

20

25

30

35

40

45

50

55

TERMINAL ~102
104 ~CARD

63   1/3

INIT FOR MUTUAL AUTHENTICATION SFI, KID, DD NUMBER, RSN ~106

RECEIVE INIT COMMAND ~108

EXTRACT DD (DIVERSIFIED DATA) FROM FILE IDENTIFIED BY THE SFI AND DD NUMBER

NOTATION:
E(•)=ENCRYPTION
D(•)=DECRYPTION
•|•=CONCATENTATION
XOR= EXCLUSIVE OR
⟹=LEADS TO CONCLUSION
(•)=OPTIONAL

GENERATE RNC ~110

RNC IS A RANDOM NUMBER GENERATED BY CARD

GENERATE CRYPTOGRAMS R1, S1, USING KD ~112

$R1=E(RNC\ XOR\ RNS)$, $S1=E(RNC\ XOR\ DD)$ WITH KD IDENTIFIED BY KID

RECEIVE KVN/DD/R1/S1 ~116

SEND KVN/DD/R1/S1 ~114

NOTE: DD IS NOT THE PLAINTEXT OF THE S1

GENERATE KD USING 1KVN AND DD ~118

TERMINAL RETRIEVE THE MASTER KEY WITH KVN TO GENERATE THE KD BEING USED BY THE CARD

DECRYPT R1, S1 WITH KD TO OBTAIN RNC' AND DD' ~120

$RNC'=D(R1)\ XOR\ RSN$, $DD'=D(S1)\ XOR\ RNC'$ WITH KD   THE CURRENT RSN IS UNAVAILABE TO ANY RE-PLAY
126 ⟹

CHECK DD=DD' ~122

IF DD'=DD THEN TERMINAL KNOWS THAT BOTH KD AND RNC ARE VALID

FIG. 1

100

GENERATE RNT ~124

CARD IS VALID BECAUSE TERMINAL GENERATES THE SAME KD
128 ⟹

GENERATE R2, S2 USING (KD XOR RNC) AS KEY ~130

$R2=E(RNT)$ $S2=E(RNT\ XOR\ IDT)$ WITH KD XOR RNC (OR KD XOR RNC/ RNC FOR DOUBLE LENGHT) AS KEY

132   NOTE: IDT=TERMINAL ID

SEND R2/S2/IDT ~132

RECEIVE R2/S2/IDT ~134

NOTE: AGAIN, IDT IS NOT THE PLAINTEXT OF THE S2

GENERATE KS=((RNC XOR RNT),(RNC=RNT)) ~144

GET RNT', IDT' WITH (KD XOR RNC) FROM R2, S2 ~136

$RNT'=D(R2)$ $IDT'=D(S2)\ XOR\ RNT'$ WITH KD XOR RNC (OR KD XOR RNC/RNC) AS KEY

CHECK IDT'=IDT ~138

IF TERMINAL ID MATCHES THEN CARD KNOWS THAT RNT IS VALID   TERMINAL IS VALID BECAUSE ⟹ TERMINAL GENERATES THE SAME KD

GENERATE KS=((RNC XOR RNT), (RNC=RNT)) ~140

KD IS THE SESSION KEY. THE INDIVIDUAL RNC RNC AND RNT SHOULD BE DESTROYED SAFETY.

MUTUAL AUTHENTICATION COMPLETE ~146

EP 1 223 565 A1

_200_

FIG. 2

27

TERMINAL (GROUND)   CARD (PORTABLE DEVICE)
302   304

                    INIT_FOR_MA   308

AUTHENTICATION
    PASS   MA   312   310
    306

                    SESSION COMMAND 1 e.g.   314
                    READ BINARY   322

                    SESSION COMMAND 2 e.g.   324
                    UPDATE BINARY   326

                    SESSION COMMAND 3 e.g.   328
                    READ RECORD   330

        320   SESSION COMMAND e.g.   332
                    UPDATE RECORD   334

                                                 336
                    SESSION COMMAND N e.g.
                    READ BINARY   340

        350   COMMIT (SESSION   342
    OMMIT   AUTHENTICATION COSE: SAC)   352
END OF SESSION   354   300

FIG. 3

                402   430   404   432   406
DURING   NVM
THE SESSION   I/O DATA   RAM CACHE   CACHE BUFFER
                                                                        408
                        420   422   FILE
                                                            SYSTEM

                                 404   440   406   442
DURING   RAM CACHE   NVM
THE COMMIT   CACHE BUFFER

        400   FILE
                                              408   SYSTEM

FIG. 4
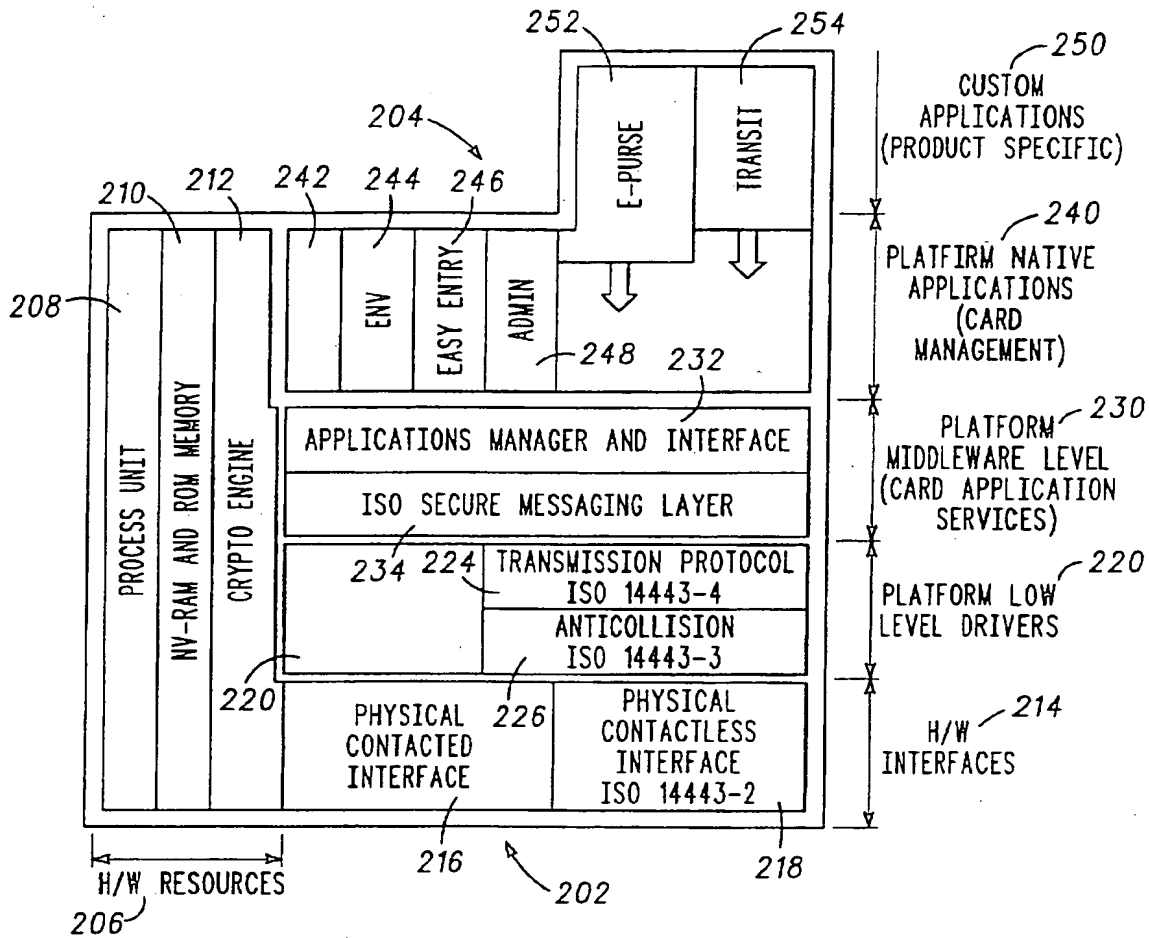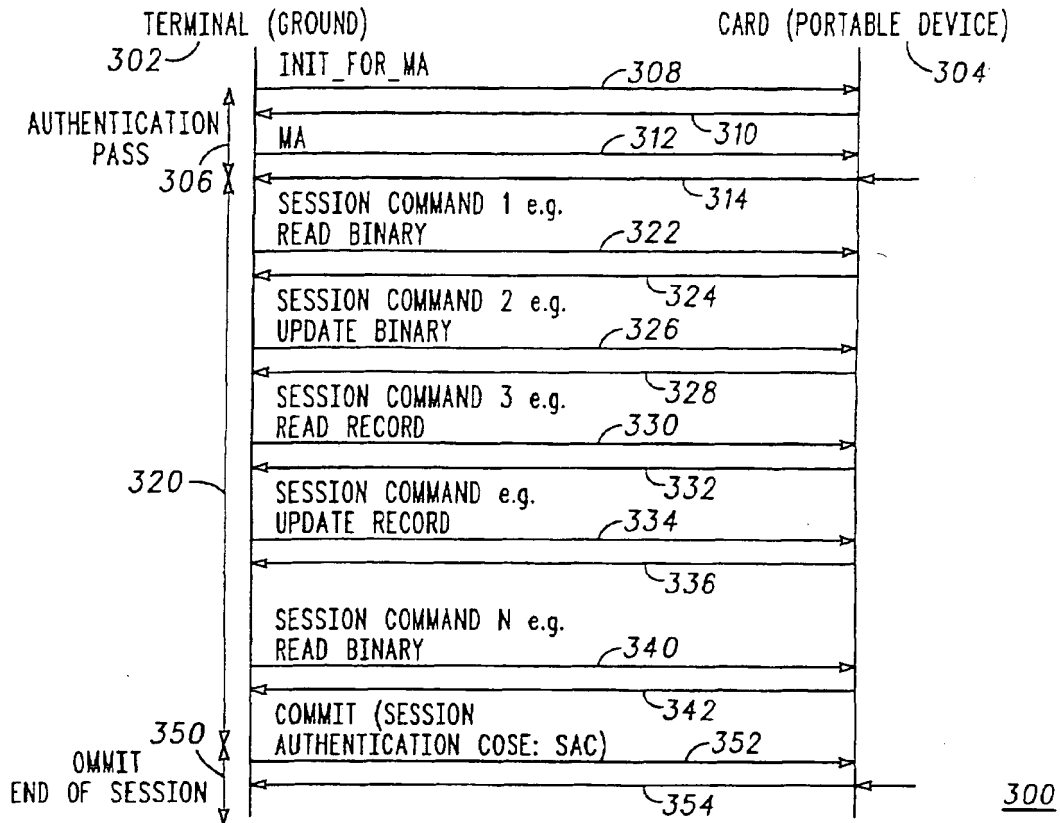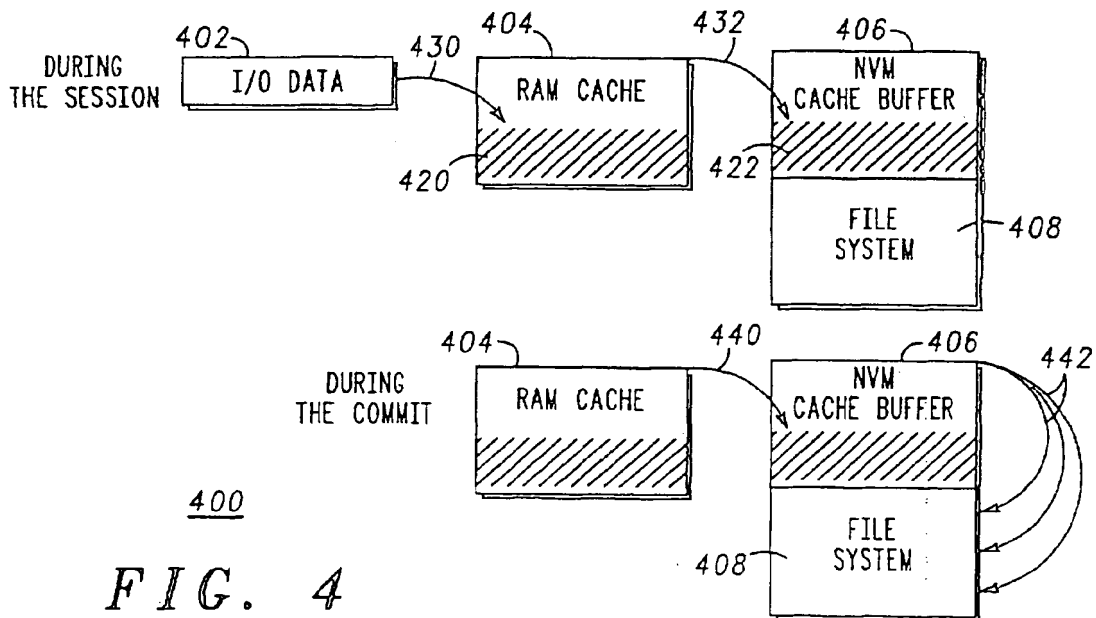
EP 1 223 565 A1

European Patent
Office

EUROPEAN SEARCH REPORT

Application Number

EP 01 40 0091

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	WO 98 48389 A (GIESECKE & DEVRIENT) 29 October 1998 (1998-10-29) * the whole document *	1,2, 4-13,28, 29	G07F7/10
A	EP 0 440 158 A (TOSHIBA) 7 August 1991 (1991-08-07) * abstract; claims; figures *	1,2,4-6, 10-13, 28,29	
A	EP 0 552 392 A (SIEMENS NIXDORF INFORMATIONSSYSTEME) 28 July 1993 (1993-07-28) * the whole document *	1,3-6. 9-15, 17-21	
A	WO 97 22091 A (KONINLKIJKE PTT NEDERLAND) 19 June 1997 (1997-06-19) * the whole document *	1,4-6, 9-13, 22-24	
A	FR 2 777 371 A (INNOVATRON ÉLECTRONIQUE) 15 October 1999 (1999-10-15) * abstract; claims *	1,9-13, 22-29	**TECHNICAL FIELDS SEARCHED** (Int.Cl.7) G07F H04L
A	EP 1 022 696 A (GIESECKE & DEVRIENT) 26 July 2000 (2000-07-26)		
A	EP 0 613 105 A (FRANCE TELECOM) 31 August 1994 (1994-08-31)		
A	EP 0 281 058 A (SIEMENS) 7 September 1988 (1988-09-07)		
A	GB 2 227 111 A (TOSHIBA) 18 July 1990 (1990-07-18)		

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	15 August 2001	David, J

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

29

ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.

EP 01 40 0091

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

15-08-2001

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
WO 9848389	A	29-10-1998	DE	19716111 A	22-10-1998
			AU	8013598 A	13-11-1998
			EP	1010146 A	21-06-2000
EP 0440158	A	07-08-1991	JP	3224080 A	03-10-1991
			JP	3224083 A	03-10-1991
			DE	69127560 D	16-10-1997
			DE	69127560 T	23-04-1998
			US	5225664 A	06-07-1993
EP 0552392	A	28-07-1993	AT	136139 T	15-04-1996
			DE	59205856 D	02-05-1996
			ES	2084846 T	16-05-1996
WO 9722091	A	19-06-1997	NL	1001863 C	10-06-1997
			AT	190420 T	15-03-2000
			AU	703985 B	01-04-1999
			AU	7625496 A	03-07-1997
			BG	102607 A	29-01-1999
			BR	9611975 A	17-02-1999
			CA	2239875 A	19-06-1997
			CN	1203681 A	30-12-1998
			CZ	9801758 A	13-01-1999
			DE	69607041 D	13-04-2000
			DE	69607041 T	13-07-2000
			EP	0883866 A	16-12-1998
			ES	2143790 T	16-05-2000
			NO	982527 A	03-06-1998
			NZ	322489 A	28-10-1998
			PL	327192 A	23-11-1998
			US	5991412 A	23-11-1999
FR 2777371	A	15-10-1999	AU	3152399 A	01-11-1999
			BR	9910116 A	26-12-2000
			CN	1296601 T	23-05-2001
			EP	1070303 A	24-01-2001
			WO	9953451 A	21-10-1999
			TR	200002933 T	21-12-2000
EP 1022696	A	26-07-2000	DE	19902722 A	27-07-2000
			JP	2000231329 A	22-08-2000
EP 0613105	A	31-08-1994	FR	2702066 A	02-09-1994
			DE	69408176 D	05-03-1998
			DE	69408176 T	30-07-1998
			US	5602915 A	11-02-1997

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.

EP 01 40 0091

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

15-08-2001

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
EP 0281058	A	07-09-1988	AT	85444 T	15-02-1993
			DE	3877984 A	18-03-1993
			ES	2041712 T	01-12-1993
			JP	63229541 A	26-09-1988
			US	4786790 A	22-11-1988
GB 2227111	A	18-07-1990	JP	2187888 A	24-07-1990
			JP	2187785 A	23-07-1990
			FR	2641885 A	20-07-1990
			HK	1003129 A	09-10-1998
			US	5293029 A	08-03-1994

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

THIS PAGE BLANK (USPTO)

THIS PAGE BLANK (USPTO)